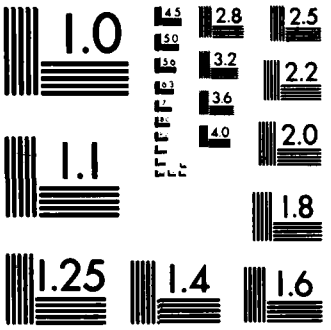
AD-A131 531 MAPPING BETWEEN SEMANTIC REPRESENTATIONS USING HORN CLAUSES(U) DELAWARE UNIV NEWARK DEPT OF COMPUTER AND INFORMATION SCIENCES R M WEISCHEDEL JUN 83 1/1

UNCLASSIFIED AFOSR-TR-83-0686 AFOSR-80-0190 F/G 5/7 NL



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(2)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-0686	2. GOVT ACCESSION NO. AD-A131531	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) "MAPPING BETWEEN SEMANTIC REPRESENTATIONS USING HORN CLAUSES"		5. TYPE OF REPORT & PERIOD COVERED Technical
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ralph M. Weischedel		8. CONTRACT OR GRANT NUMBER(s) AFOSR-80-0190
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Delaware Computer & Information Sciences Newark, DE 19711		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS AFOSR/NM Building 410 Bolling AFB DC 20332		12. REPORT DATE June 1983
		13. NUMBER OF PAGES 8
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Even after an unambiguous semantic interpretation has been computed for a sentence in context, there are at least three reasons that a system may map the semantic representation R into another form S:
1) The terms of R. while reflecting the user view, may require deeper understanding, e.g. may require a version S where metaphors have been analyzed.
2) Transformations of R may be more appropriate for the underlying application system, e.g. S may be a more nearly optimal form. These transformations may not
CONT.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered

be linguisticly motivated.
3) Some transformations may depend on non-structural context.

Design considerations may favor factoring the process into two stages, for reasons of understandability or for easier transportability of the components.

This paper describes the use of Horn clauses for the three classes of transformations listed above. The transformations are part of a system that converts the English description of a software module into a formal specification, i.e. an abstract data type.

MAPPING BETWEEN SEMANTIC REPRESENTATIONS USING HORN CLAUSES[1]

Ralph M. Weischedel
Computer & Information Sciences
University of Delaware
Newark, DE 19711

For consideration in NATURAL LANGUAGE

ABSTRACT

Even after an unambiguous semantic interpretation has been computed for a sentence in context, there are at least three reasons that a system may map the semantic representation R into another form S.

1. The terms of R, while reflecting the user view, may require deeper understanding, e.g. may require a version S where metaphors have been analyzed.

2. Transformations of R may be more appropriate for the underlying application system, e.g. S may be a more nearly optimal form. These transformations may not be linguisticly motivated.

3. Some transformations may depend on non-structural context.

Design considerations may favor factoring the process into two stages, for reasons of understandability or for easier transportability of the components.

This paper describes the use of Horn clauses for the three classes of transformations listed above. The transformations are part of a system that converts the English description of a software module into a formal specification, i.e. an abstract data type.

1. INTRODUCTION

Parsing, semantic interpretation, definite reference resolution, quantifier scope decisions, and determining the intent of a speaker/author are well-known problems of natural langauge understanding. Yet, even after a system has generated a semantic representation R where such decisions have been made, there may still be a need for further transformation and understanding of the input to generate a representation S for the underlying application system. There are at least three reasons for this.

First, consider spatial metaphor. Understanding spatial metaphor seems to require

---

[1] RESEARCH SPONSORED BY THE AIR FORCE OFFICE OF SCIENTIFIC RESEARCH, AIR FORCE SYSTEM COMMAND, USAF, UNDER GRANT NUMBER AFOSR-80-0190. THE UNITED STATES GOVERNMENT IS AUTHORIZED TO REPRODUCE AND DISTRIBUTE REPRINTS FOR GOVERNMENTAL PUPOSES NOTWITHSTANDING ANY COPYRIGHT NOTATION HEREIN

computing some concrete interpretation S for the metaphor; however, understanding the metaphor concretely may be attempted after computing a semantic representation R that represents the spatial metaphor formally but without full understanding. Explaining the system's interpretation of a user input (e.g. for clarification dialog, allowing the user to check the system's undersanding, etc.) is likely to be more understandable if the terminology of the user is employed. By having an intermediate level of understanding such as R, and generating English output from it, one may not have to recreate the metaphor, for the terms in R use it as a primitive.

Second, the needs of the underlying application system may dictate transformations that are neither essential to understanding the English text nor linguisticly motivated. In a data base environment, transformations of the semantic representation may yield a retrieval request that is computationally less demanding [King 80]. To promote portability, EUFID [Templeton 83] and TQA [Damerau 81] are interfaces that have a separate component for transformations specific to the data base. In software specification, mapping of the semantic representation R may yield a form S which is more amenable for proving theorems about the specification or for rewriting it into some standard form.

The following example, derived from a definition of stacks on page 77 of [Horowitz 76] illustrates both of the reasons above.

> A stack is an ordered list in which all insertions and deletions occur at one end called the top.

A theorem prover for abstract data types would normally assume that the end of the stack in question is referred to by a notation such as A[1] if A is the name of the stack, rather than understanding the spatial metaphor "one end".

Third, it may be convenient to design the transformation process in two phases, where the output of both phases is a semantic representation. In our system, we have chosen to map certain paraphrases into a common form via a two step process. The forms "ith element" and "element i" each generate the same term as a result of semantic interpretation. However, the semantic interpreter generates another term for "element at position i" due to the extra lexical items "at" and "position". Obviously, all three expressions correspond to one concept. The mapping component recognizes that the two terms generated by the semantic interpreter are paraphrases and maps them into one form.

Section 2 gives an overview of the system as a whole. Section 3 describes the use of Horn clauses for the mapping from R to S. Related research and our conclusions are presented in sections 4 and 5.

2. BRIEF SYSTEM OVERVIEW

The overall system contains several components beside the mapping component that is the focus of this paper. The system takes as input short English texts such as the data structure descriptions in [Horowitz 76]. The output is a formal specification of the data structure defined in Horn clauses[2]. [In a full version of the paper, a sample text of about 10 sentences and the Horn clause output will appear here.]

First, the RUS parser [Bobrow 78], which includes a large general-purpose grammar of English, calls a semantic component to incrementally compute the semantic interpretation of the phrases being proposed. As soon as a phrase is proposed by the grammar, the semantic interpreter either generates a semantic interpretation for the phrase or vetoes the parse. The only modifications to adapt the parser to the application of abstract data types were to add mathematical notation, so that phrases such as "the list (A[1], A[2], ..., A[N])" could be understood. The semantic component we developed employs case frames for disambiguation and generation of the semantic interpretation of a phrase. However, the semantic component does not make quantifier scope decisions.

Quantifier scope decisions, reference resolution, and conversion from first-order logic to Horn clauses is performed after the semantic interpreter has completed its processing. The knowledge governing these three tasks is itself encoded in Horn clauses and was developed by Daniel Chester. The output from this component is the input to the mapping component, which is the focus of this paper.

The semantic representation R of a single sentence is therefore a set of Horn clauses. In addition, the model of context built in understanding the text up to the current sentence is a set of Horn clauses and a list of entities which could be referenced in succeeding sentences. The mapping component performs the three tasks described in the previous section to generate a set S of Horn clauses. S is added to the model of context prior to processing the next input sentence.

The choice of Horn clauses as the formal representation of the abstract data type is based on the following motivations:

1. Once a text has been understood, the set of Horn clauses can be added to the knowledge base (which is also encoded as Horn clauses). This offers the potential of a system that grows in power.

2. The semantics of Horn clauses, their use in theorem proving, and their executability makes them an appropriate formalism for defining abstract data types.

3. A Horn clause theorem prover [Chester 80] allowing free intermixing of lisp and

---

[2] Horn clauses are a version of first order logic, where all well-formed formulas have the form A1 & A2 & ... & An -> C. Each of the Ai is an atomic formula; C is an atomic formula; and n>=0. Therefore, all variables are free.

theorem proving is readily available.

3. MAPPING IN THE SYSTEM

The rules of the mapping component are all encoded as Horn clauses. The antecedent atomic formulas of our rules specify either

1. the structural change to be made in the collection of formulas or
2. conditions which are not structural in nature but which must be true if the mapping is to apply.

We will use the notation (MAPPING-RULE (a1 ... am) ?x (c1 ... ck) ?y) to mean that the atomic formulas a1 ... am must be present in the list ?x of atomic formulas; the list ?x of formulas is assumed to be implicitly conjoined. The variable ?y will be bound to the result of replacing the formulas a1, ..., am in ?x with the formulas c1, ..., ck. There is a map between two lists, ?x and ?y, of atomic formulas if (MAP ?x ?y) is true.

The two examples given in the introduction are detailed next. For expository purposes the rules given in this section have been simplified.

Consider the following example:

A stack is an ordered list in which all insertions and deletions occur at one end called the top. ADD(I,S) adds item I to stack S.

In this environment spatial metaphors tend to be more frozen than creative. To understand "one end", we assume the following rules:

1. For a sequence ?D, we may map "?E is an end of ?D" to "?E is a the first sequence element of ?D".
2. An ordered list is a sequence.

Facts (1) and (2) are encoded as Horn clauses below.

1. (SEQUENCE ?D) (MAPPING-RULE ((END ?E ?D)) ?X
   ((SEQUENCE-ELEMENT ?E 1 ?D)) ?Y)
   => (MAP ?X ?Y)

2. (ORDERED-LIST ?D) => (SEQUENCE ?D)

The system knows how to map the notion of "end of a sequence", and it knows that ordered lists are sequences. Since the first sentence is discussing the end of an ordered list, the two rules above are sufficient to map "end" into the appropriate concrete semantic representation. The power and generality of this approach is that

- a chain of reasoning may show how to view some entity ?D as a sequence (and therfore the rules show how to interpret "end of ?D"), and
- other mapping rules may state how to interpret spatial metaphors unrelated to "end"

or to sequences.

[A long version of the paper can give rules for how the sense of "add" being defined in the example above can be interpreted in this environment.]

The second example from the introduction involves mapping the forms "ith element", "element i", and "element at position i" into the same representation. Assume that the semantic interpreter generates for each of the first two the list of formulas ((ELEMENT ?X) (IDENTIFIED-BY ?X ?Y)). The Horn clause for that mapping is as follows:

```
(SEQUENCE ?T) (TOPIC ?T)
(MAPPING-RULE ((ELEMENT ?X) (IDENTIFIED-BY ?X ?Y)) ?W
              ((SEQUENCE-ELEMENT ?X ?Y ?T)) ?Z)
   => (MAP ?W ?Z)
```

Note that this rule assumes that in context some sequence ?T has been identified as the topic; the rule identifies that the element ?X is the ?Yth member of the sequence ?T. For the phrase "element at position i", assume the semantic interpreter generates the list of formulas ((ELEMENT ?X) (AT ?X (POSITION ?P)) (IDENTIFIED-BY ?P ?Y)). The mapping rule for it is similar to the one above.

```
(SEQUENCE ?T) (TOPIC ?T)
(MAPPING-RULE ((ELEMENT ?X) (AT ?X (POSITION ?P)) (IDENTIFIED-BY ?P ?Y)) ?W
              ((SEQUENCE-ELEMENT ?X ?Y ?T)) ?Z)
   => (MAP ?W ?Z)
```

This second rule must be tried before the prior one.

The mapper halts when no more rules can be applied.

4. RELATED WORK

A number of applied AI systems have been developed to support automating software construction [Balzer 78, Green 76, Biermann 80, Gomez 82]. Of these, only our effort has focussed on the mapping problem.

Viewing spatial metaphors in terms of a scale was proposed in [Hobbs 77]. Our model is somewhat more general in that the inference process

- permits specific constraints for each metaphor, not just the one view of a scale, and
- accounts for other mapping problems in addition to spatial metaphor.

A very similar approah to mapping has been proposed in [Mark 80]. Instead of using Horn clauses as the formalism for mapping, they encode their rules in KL-ONE [Brachman 78]. The concern in [Mark 80] is inferring the appropriate service to perform in response to

a user request, rather than demonstrating means of interpreting spatial metaphors or of finding contextually dependent paraphrases.

5. CONCLUSIONS

There are several reasons why one may have a second transduction phase even after a semantic representation for an utterance has been computed. The advantage of using Horn clauses (or any other deduction mechanism) in this mapping phase is the ability to include nonstructural conditions. This means that the mapping rules may be based on reasoning about the context.

There are three areas for future work:

- generating mapping rules based on additional texts,
- investigating use of the mapping component in reference resolution, and
- developing an indexing technique to run the mapper in a forward chaining mode.

REFERENCES

[Balzer 78] Robert Balzer, Neil Goldman, and David Wile.
Informality in Program Specification.
IEEE Transactions on Software Engineering SE-4(2), March, 1978.

[Biermann 80] Alan W. Biermann and Bruce W. Ballard.
Toward Natural Language Computation.
JACL 6(2), 1980.

[Bobrow 78] R.J. Bobrow.
The RUS System.
In B.L. Webber, R. Bobrow (editors), *Research in Natural Language Understanding*, . Bolt, Beranek and Newman, Inc., Cambridge, MA, 1978.
BBN Technical Report 3878.

[Brachman 78] Ronald Brachman.
A Structural Paradigm for Representing Knowledge.
Technical Report, Bolt, Beranek, and Newman, Inc., 1978.

[Chester 80] Daniel L. Chester.
HCPRVR: An Interpreter for Logic Programs.
In *Proceedings of the National Conference for Artificial Intelligence*, pages 93-95. American Asociation for Artificial Intelligence, Aug, 1980.

[Damerau 81] Fred J. Damerau.
Operating Statistics for the Transformational Question Answering System.
American Journal of Computational Linguistics 7(1):30-42, 1981.

[Gomez 82] Fernando Gomez.
Towards a Theory of Comprehension of Declarative Contexts.
In *Proceedings of the 20th Annual Meeting of the Association for Computational Linguisitcs*, pages 36-43. Association for Computational Linguistics, June, 1982.

[Green 76] C. Green.
The Design of the PSI Program Synthesis System.
In *Second International Conference on Software Engineering*. IEEE Computer Society, Oct, 1976.

[Hobbs 77] Jerry R. Hobbs.
What the Nature of Natural Language Tells us about how to Make Natural-Langauge-Like Programming More Natural.
In *SIGPLAN Notices*, pages 85-93. SIGPLAN, 1977.

[Horowitz 76] Ellis Horowitz and Sartaj Sahni.
Fundamentals of Data Structures.
Computer Science Press, Woodland Hills, CA, 1976.

[King 80] Jonathan J. King.
Intelligent Retrieval Planning.
In *Proceedings of the National Conference for Artificial Intelligence*, pages 243-245. American Asociation for Artificial Intelligence, Aug, 1980.

[Mark 80] William Mark.
Rule-Based Inference In Large Knowledge Bases.
In *Proceedings of the National Conference on Artificial Intelligence.* American Association for Artificial Intelligence, August, 1980.

[Templeton 83]
Marjorie Templeton and John Burger.
Problems in Natural Language Interface fo DBMS with Examples from EUFID.
In *Conference on Natural Language Processing*, pages 3-16. Association for Computational Linguistics, Feb, 1983.

END

FILMED

9-83

DTIC